# What came first, the kernel or the shell?

Alex Gantman · Following
7 min read · Mar 29, 2023



If your work is even remotely related to operating systems, the terms *shell* and *kernel* are likely in your everyday vocabulary. But where, and when, did they appear first?

## The Metaphors

The *kernel* is the core component of the operating system that executes in a more privileged mode and provides security isolation between processes. The *shell* is the operating system's interface to the user, a kind of wrapper around the kernel. This framing implies a natural synergy between the two metaphors.


Huang, S.-L.; Wang, W.-H.; Zhong, X.-Y.; Lin, C.-T.; Lin, W.-S.; Chang, M.-Y.; Lin, Y.-S. Antioxidant Properties of Jatropha curcas L. Seed Shell and Kernel Extracts. Appl. Sci. 2020, 10, 3279. https://doi.org/10.3390/app10093279

As it turns out, the apparent hand-in-glove match is coincidental. The two terms evolved independently of each other[1], almost a decade apart.

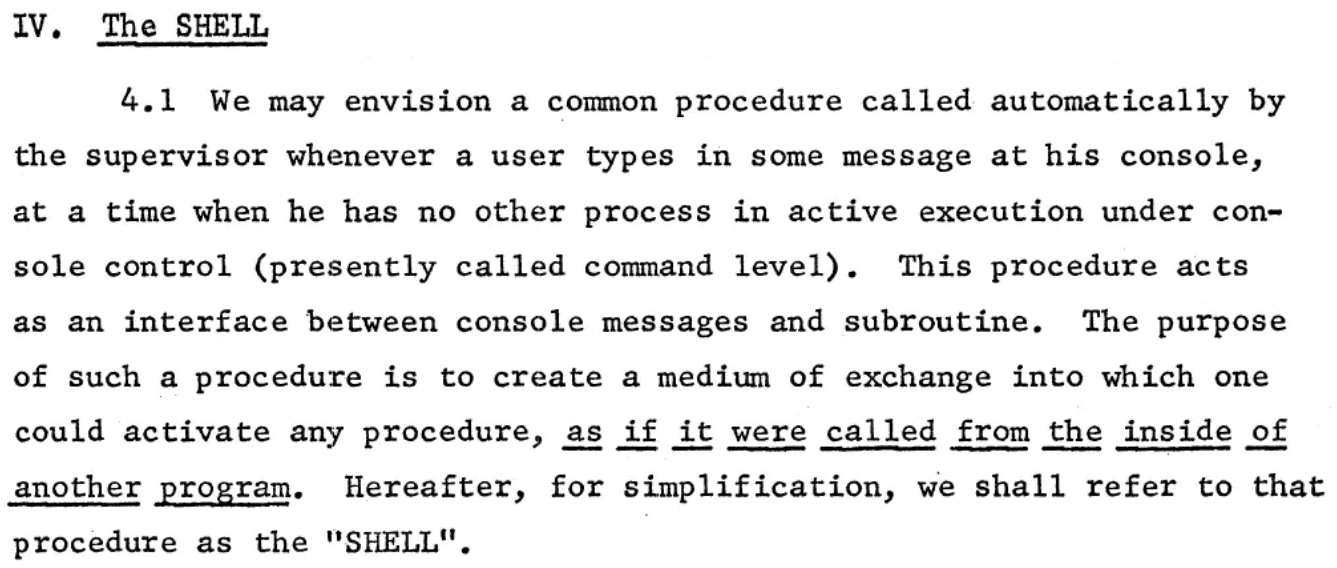
## The Shell

The concept of the shell as a service that could interpret and run commands form the user was first implemented by Louis Pouzin in 1963 in a system named RUNCOM.

> *I wrote "RUNCOM", a sort of shell driving the execution of command scripts, with argument substitution.*
> *[https://multicians.org/shell.html]*

The term *shell* was also introduced by Pouzin about a year later, when he released the MULTICS SHELL.

> *Then in 64 came the Multics design time, [...] I wrote a paper explaining how the Multics command language could be designed with this objective. And I coined the word "shell" to name it. It must have been at the end of 64 or beginning of 65.*
> *[https://multicians.org/shell.html]*



IV. The SHELL

4.1 We may envision a common procedure called automatically by the supervisor whenever a user types in some message at his console, at a time when he has no other process in active execution under console control (presently called command level). This procedure acts as an interface between console messages and subroutine. The purpose of such a procedure is to create a medium of exchange into which one could activate any procedure, as if it were called from the inside of another program. Hereafter, for simplification, we shall refer to that procedure as the "SHELL".

Pouzin, L. The SHELL: A Global Tool for Calling and Chaining Procedures in the System, 1965
https://people.csail.mit.edu/saltzer/Multics/Multics-Documents/MDN/MDN-4.pdf

As a metaphor, Pouzin's SHELL was really more a wrapper around the invoked command than an outer layer of the operating system.

When Ritchie and Thompson started writing UNIX in 1969, they adopted the terminology from MULTICS.

5. The Shell

5.1 General

Communication with UNIX is carried on with the aid of a program called the Shell. The Shell is a command line interpreter: it reads lines typed by the user and interprets them as requests to execute other programs. In simplest form, a command line consists of the command name followed by arguments to the command, all

Ritchie, D. M. The UNIX Time-Sharing System (unpublished draft), 1971
https://www.tuhs.org/Archive/Distributions/Research/McIlroy_v0/UnixEditionZero-Threshold_OCR.pdf

## The Kernel

It turns out that the kernel has a more convoluted history. Before we could have the kernel, we needed to have hardware support for modes of privileged execution and the abstraction of a resource manager.

### Privileged Execution

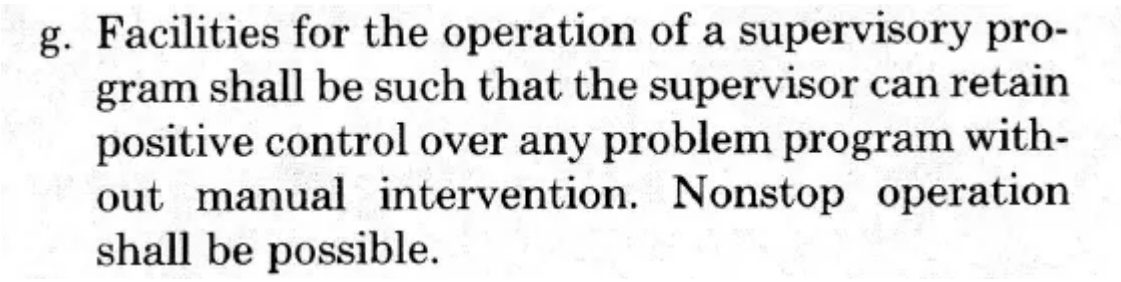
Hardware support for an operating system executing in a more privileged state than user programs dates back to at least 1961 and the Ferranti Atlas computer.

## Supervisor Extracode Routines

Supervisor extracode routines (S.E.R.'s) form the principal "branches" of the supervisor program. They are activated either by interrupt routines or by extracode instructions occurring in an object program. They are protected from interference by object programs by using subsidiary store as working space, together with areas of core and drum store which are locked out in the usual way whilst an object program is being executed (see Section 3). They operate under extracode control, the extracode control register of any current object program being preserved and subsequently restored. Like the interrupt routines, they use private B-lines, in this case B-lines 100 to 110 inclusive; if any other working registers are required, the supervisory routines themselves preserve and subsequently restore the contents of such registers. The S.E.R.'s thus apply mutual protection between themselves and an object program.

T. Kilburn, R. B. Payne, and D. J. Howarth. 1961. The Atlas supervisor. In Proceedings of the December 12–14, 1961, eastern joint computer conference: computers — key to total systems control (AFIPS '61 (Eastern)). Association for Computing Machinery, New York, NY, USA, 279–294. https://doi.org/10.1145/1460764.146078

The importance of having a more privileged mode for the operating system was also recognized in the 1961 IBM SPREAD report, that led to the creation of System/360.



g. Facilities for the operation of a supervisory program shall be such that the supervisor can retain positive control over any problem program without manual intervention. Nonstop operation shall be possible.

J. W. Haanstra et al., "Processor Products-Final Report of the SPREAD Task Group, December 28, 1961," in Annals of the History of Computing, vol. 5, no. 1, pp. 6–26, Jan.-March 1983, doi: 10.1109/MAHC.1983.10007

When System/360 hardware appeared in 1964, it supported System and Problem states, for supervisor and user code, respectively. Similarly, the GE-635 that came out at the same time implemented Master and Slave modes to distinguish privileged from unprivileged software.

**Dual-Mode Operation**

Dual mode operation is essential in a multiprogramming environment because only the operating system should be allowed to initiate input/output operations, set the interval timer, BAR, and other control registers. Without dual-mode operation, multiprogramming would require extreme discipline by the user to insure that he did not assume operating system functions.

Each Processor is capable of operating in two modes: the Slave mode or the Master mode. When a Processor is in the Master mode, no restrictions are imposed on a program. When a Processor is in the Slave mode, a program is inhibited from exercising a control relationship with Memory modules and associated peripheral devices. Also, while in the Slave mode, the Base Address Register (see below) controls relative addressing and memory limits protection for the program.

General Electric GE-635 System Manual, rev. July 1964 http://www.bitsavers.org/pdf/ge/GE-6xx/CPB-371A_GE-635_System_Man_196407.pdf

### The Reference Monitor

The concept of a reference monitor — an abstract mechanism for mediating access to resources — was first published by Peter Denning and Scott Graham in 1972.

> *When Scott [Graham] finished his thesis, we wrote a paper about his main findings for the 1972 Spring Joint Computer Conference. One aspect of that paper stuck in people's minds — the notion we called reference monitor.*
> *The reference monitor was the notion that every class of objects had a system that managed it, and that system enforced all the access rules registered in the access matrix.*
> *[https://conservancy.umn.edu/handle/11299/156515]*

Associated with each type of object is a *monitor*, through which all access to objects of that type must pass to be validated. Examples of monitors are the file system for files, the hardware for instruction execution and memory addressing, and the protection system for subjects. An access proceeds as follows:

G. Scott Graham and Peter J. Denning. 1971. Protection: principles and practice. In Proceedings of the May 16–18, 1972, spring joint computer conference (AFIPS '72 (Spring)). Association for Computing Machinery, New York, NY, USA, 417–429. https://doi.org/10.1145/1478873.1478928

Denning and Graham were the first to publish the idea, but its origin is generally credited to Roger Schell, who at the time was the program manager for the seminal Computer Security Technology Planning Study, aka the Anderson Report.

The contributions and encouragement of the program manager, Major Roger Schell, USAF (ESD/MCI) are gratefully acknowledged.

Anderson, J. Computer Security Technology Planning Study, ESD-TR-73–51, US Air Force Electronic Systems Division. https://apps.dtic.mil/sti/citations/AD0758206

> *I could have something that actually is the thing that sits in the middle of access between subjects and objects, and that's what Butler Lampson had talked about in his access matrix paper. And so I said well, okay, we'll call that the reference monitor. And then, I said well, okay, the security kernel is sort of the implementation of this abstraction of the reference monitor. And, I said, well what are the properties of the security kernel? One of the panel members asked. I suppose one of the academic ones; maybe Glaser or somebody. I hadn't thought about that. And so well, okay, let's think about it. What does it have to be. Well, it has to be always invoked, i.e., non-bypassable, so that I can't get at the objects except through the reference monitor. And it has to be small and simple enough that I can verify what it does; I have to be able to validate that it is complete. In other words, it has to review my whole policy; it has to have both positives and negatives of the policy; what I can do and what I can't do. And to be complete it has to be tamperproof. [https://conservancy.umn.edu/handle/11299/133439]*

Denning acknowledges Schell in his oral history.

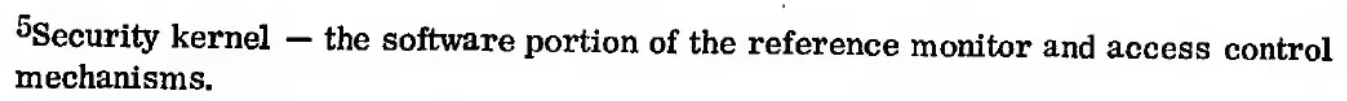
> *Denning: Right. I'm sure Roger [Schell] already figured out the necessity of the principle before Graham and I wrote about it. To me, at the time when we wrote the paper, it was like an obvious thing. […] It seemed to me that we were just recording common wisdom. Maybe we were the first to put a name on it. Process managers, virtual memory managers, and file managers in operating systems all worked this way. […] Graham and I did not think of reference monitor as the main contribution of the paper;*
> *[https://conservancy.umn.edu/handle/11299/133439]*

**The Colonel and the Kernel**

Though Schell came up with the concept, he needed help picking a good name for it. He recalls that the term "*kernel*" was suggested by his manager, John Goodenough.

> *So [Goodenough] says, well you know, what you describe sort of sounds like what in mathematics what we think of as a kernel. […] And so we kicked around ideas and it's about security, let's call it a "Security Kernel." So, yes, he was the one who created the name of Security Kernel so that I'd have something to write down.*
> *[https://conservancy.umn.edu/handle/11299/133439]*

When the Anderson Report came out in 1972[2] it had embraced both the concept and the term.


[5]Security kernel — the software portion of the reference monitor and access control mechanisms.

Anderson, J. Computer Security Technology Planning Study, ESD-TR-73–51, US Air Force Electronic Systems Division. https://apps.dtic.mil/sti/citations/AD0758206

As for Schell, he eventually reached the rank of Colonel before retiring from the Air Force. I hope Dr Schell will forgive my childish indulgence, but that means that the kernel was invented by Colonel Schell [pronounced "kernel shell"], Ret.

**An Unexplained Oddity**

Curiously, the 1971 PDP11/45 Processor Handbook also adopts the term *kernel* to describe the more privileged mode of execution.[3]

The machine operates in three modes: Kernel, Supervisor, and User. When the machine is in Kernel mode a program has complete control of the machine; when the machine is in any other mode the processor is inhibited from executing certain instructions and can be denied direct access to the peripherals on the system. This hardware feature can be used to provide complete executive protection in a multi-programming environment.

Digital Equipment Corporation PDP11/45 Processor Handbook, 1971
https://www.computerhistory.org/collections/catalog/102683353

This is mysterious because it predates by a year any published reference to *kernel* from any of the Anderson Study participants. I also could not find any connection between the Anderson Study and DEC. The usage of *kernel* within PDP11/45 designs means that either there was an undocumented interaction between DEC designers and the Anderson Study, or two separate teams came up with the term independently at around the same time.

## UNIX, the Kernel, and the Shell

It's probably safe to say that in most people's minds there is no UNIX without the UNIX kernel. Yet, although the shell was mentioned in UNIX papers from the very first draft in 1971, the word kernel did not make its appearance until 1978 (in a revision of a 1974 paper), and even then, only in passing.

Nonetheless, by 1975 the two terms were appearing side by side in operating systems literature (quite possibly due to popularity of the new PDP11/45 and its descendants) and quickly became essential parts of the technical lexicon.

2.1 Unix

The operating system uses the standard DEC address relocation hardware to partition the physical memory into two parts - kernel space which is reserved for the resident portion of Unix, and user space which is available to user programs. The user's keyboard interface to the system is a program called the shell. The system

Gregory L. Chesson. 1975. The network Unix system. SIGOPS Oper. Syst. Rev. 9, 5 (November 1975), 60–66.
https://doi.org/10.1145/1067629.806522

And now you know.


## Acknowledgements

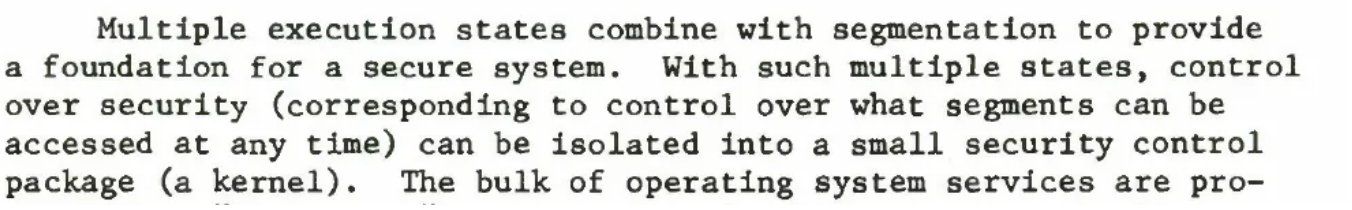
I am indebted to Steve Lipner for many personal recollections, pointers, and corrections to wildly incorrect claims I kept trying to make. Obviously, all remaining mistakes are entirely mine.

I am also grateful to Dag Spicer and Penny Ahlstrand of the Computer History Museum for getting the scans of the 1971 edition of the PDP11/45 Processor Handbook.


## Notes

[1] However, it is entirely possible that even though the terms appeared independently, the reason they stuck was the mutual fit of the metaphors.

[2] Steve Lipner's 1972 SATIN paper also adopted kernel terminology and came out about a month before the Anderson Report. But since Lipner participated in the Anderson Study, the term had the same source.



Multiple execution states combine with segmentation to provide a foundation for a secure system. With such multiple states, control over security (corresponding to control over what segments can be accessed at any time) can be isolated into a small security control package (a kernel). The bulk of operating system services are pro-

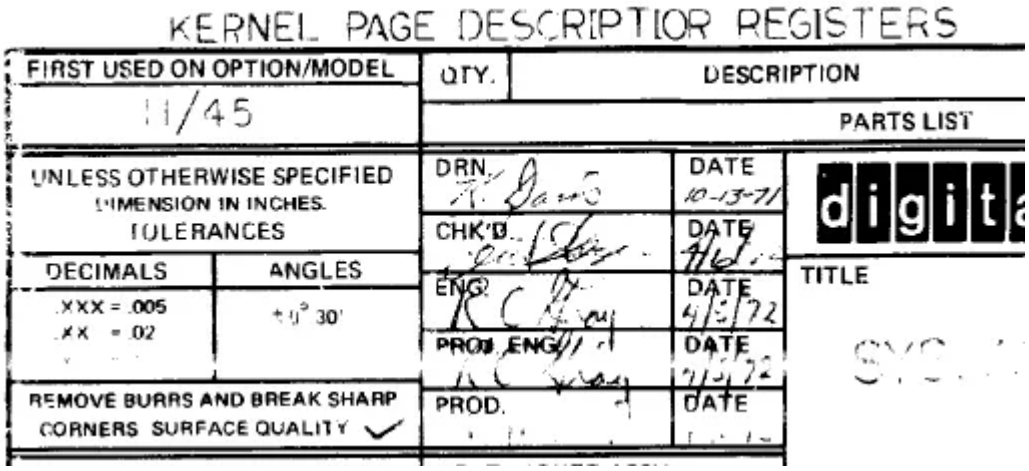
Lipner, S. SATIN Computer Security, Air Force Electronic Systems Division document MCI-75–2, September 1972
https://apps.dtic.mil/sti/citations/ADA542784

[3] There is also the following reference in an internal DEC memo from October 1971:

Attempting to rush the design and implementation of the OS/45 Kernel in order to permit DOS & RSTS to

Peter van Roekens, OS/45 proposal Meeting, Digital Equipment Corporation Internal Memorandum, October 20, 1971, https://bitsavers.org/pdf/dec/pdp11/memos/711020_OS45_Proposal_Meeting.pdf

As well as the 1972 engineering drawings for both, the KB11-A CPU and the KT11-C MMU from PDP11/45 (pay attention to the dates and signatures on individual pages, not the overall publication date).


KERNEL PAGE DESCRIPTIOR REGISTERS

| FIRST USED ON OPTION/MODEL | QTY. | DESCRIPTION |
|---|---|---|
| 11/45 | | PARTS LIST |

| UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES | | | | |
|---|---|---|---|---|
| DECIMALS | ANGLES | DRN. | DATE 10-13-71 | digita |
| .XXX = .005 .XX = .02 | ±0° 30' | CHK'D. | DATE | TITLE |
| | | ENG. | DATE 4/5/72 | |
| | | PROJ. ENG. | DATE | |
| REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓ | | PROD. | DATE | |

Digital Equipment Corporation KT11-C Memory Management Unit Engineering Drawings, 1972, https://bitsavers.org/pdf/dec/pdp11/1145/KT11C_Schem.pdf

Cross-posted to LinkedIn.

Computer History